Simulation of Pile-Up Effects

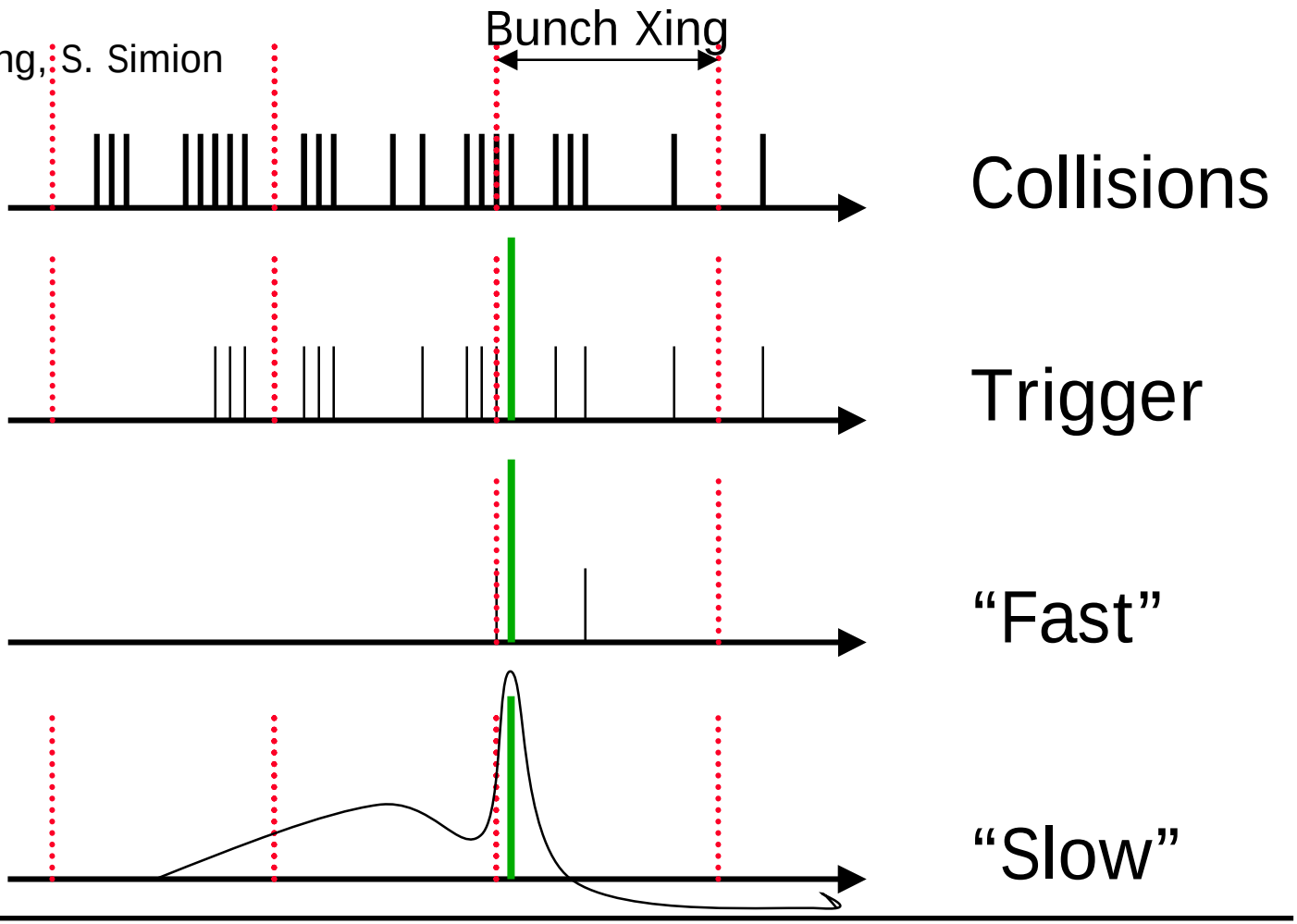
Paolo Calafiura, LBL

Architecture Workshop

January 23 2001

What is it (my understanding)

- Physics TDR 2.3
- e-mails F. Luehring, S. Simion



TDR Strategies

- Standard - Production
  - estimate “effective” #Interactions in sub-detector sensitive time (including out-of-time contributions)
  - overlay hits from #Interactions min bias G3 events to the physics event hits (no hits time info)
  - digitize
- Complete - TRT studies
  - keep track of simulated hit times
  - build a time history around the physics trigger
  - collect hits in a sub-detector specific time window
  - digitize
- Pulse Shape - LAr studies
  - Use individual pulse shapes to merge hit contributions

Some Numbers

- Typical TDR job
  - Average 23 collision per bunch crossing
    - 3500 Particles at |eta|<5
    - 2-4 bunch crossings
  - TDR min bias events 2MByte
    - special studies ranged 0.3-4MB/evt
  - up to 200 overlaid events (allowing for tails)
- Worst case scenario
  - >~800 min bias events (hi-luminosity)
  - up to 4MB/event (lower Geant cutoff)
- 1-4GB total memory usage
  - TDR production brute-force approach (all evts in memory)
  - read in and overlay one sub-detector after another?

Some Functional Requirements

- overlaid evts generated merging hits before digitization
  - hits will come from at least two streams (physics and min bias)
  - min bias stream must allow random access
- pile-up is performed bunch-crossing by bunch-crossing
  - particle time of flights are optionally taken into account in a time history for the crossing
  - the number of min bias events in each bunch-crossing can be set or selected at random according to a Poisson
    - this can be done subdetector by subdetector (effectively allowing to use the TDR "standard method")
- it should be possible to overlay each subdetector (sector?) separately - memory management
- possible to determine which hit from which track from which event contributed to a digitization

How it might look like

```
ApplicationMgr.TopAlg = {"EventOverlay/MyMBiasOvrl", ...};
MyMBiasOvrl.Mergers = {"MergePixel", "MergeTRT"...}; //subalg?
MyMBiasOvrl.physicsStreams = {"PhysicsSelector"};
MyMBiasOvrl.minBiasStreams = {"G3MBiasSelector"};
MyMBiasOvrl.crossingFrequency = 25 //ns
//beam time history: which mbias evts to use at which time
MyMBiasOvrl.makeCrossings = {-2, +1} //4 x-ings, -50<t<50ns
MyMBiasOvrl.collisions = 23; //per x-ing
MyMBiasOvrl.minBiasDistribution = "Poisson"; // or "Fixed"
//use "standard method" for MergePixel
MergePixel.useCrossings = {0, 0};   //default use all
MergePixel.timeGranularity = "crossing" //default "collision"
```

How would it work?

- I don't know
- Athena supports multiple input streams traversed by their own selectors
- I don't know how easily these selectors can be associated with multiple stores accessible from the merging algorithms
- The Gaudi folks mention in their guide a PileUpSvc.
  - I am not even sure what that service would be used for

EventInfo related classes